Linear Codes, Target Function Classes, and Network Computing Capacity*

Rathinakumar Appuswamy, Massimo Franceschetti,
Nikhil Karamchandani, and Kenneth Zeger

Abstract

We study the use of linear codes for network computing in single-receiver networks with various classes of target functions of the source messages. Such classes include reducible, injective, semi-injective, and linear target functions over finite fields. Computing capacity bounds and achievability are given with respect to these target function classes for network codes that use routing, linear coding, or nonlinear coding.

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407. (rathnam@ucsd.edu, massimo@ece.ucsd.edu, nikhil@ucsd.edu, zeger@ucsd.edu)

1   Introduction

Network coding concerns networks where each receiver demands a subset of messages generated by the source nodes and the objective is to satisfy the receiver demands at the maximum possible throughput rate. Accordingly, research efforts have studied coding gains over routing [1, 10, 11], whether linear codes are sufficient to achieve the capacity [6, 7, 14, 16], and cut-set upper bounds on the capacity and the tightness of such bounds [10, 11, 23].

Network computing, on the other hand, considers a more general problem in which each receiver node demands a target function of the source messages [4, 8, 15, 17, 21, 22]. Most problems in network coding are applicable to network computing as well. Network computing problems arise in various networks including sensor networks and vehicular networks.

In [4], a network computing model was proposed where the network is modeled by a directed, acyclic graph with independent, noiseless links. The sources generate independent messages and a single receiver node computes a target function f of these messages. The objective is to characterize the maximum rate of computation, that is, the maximum number of times f can be computed per network usage. Each node in the network sends out symbols on its out-edges which are arbitrary, but fixed, functions of the symbols received on its in-edges and any messages generated at the node. In linear network computing, this encoding is restricted to be linear operations. Existing techniques for computing in networks use routing, where the codeword sent out by a node consists of symbols either received by that node, or generated by the node if it is a source (e.g. [19]).

In network coding, it is known that linear codes are sufficient to achieve the coding capacity for multicast networks [1], but they are not sufficient in general to achieve the coding capacity for non-multicast networks [6]. In network computing, it is known that when multiple receiver nodes demand a scalar linear target function of the source messages, linear network codes may not be sufficient in general for solvability [20]. However, it has been shown that for single-receiver networks, linear coding is sufficient for solvability when computing a scalar linear target function [3, 21]. Analogous to the coding capacity for network coding, the notion of computing capacity was defined for network computing in [8] and is the supremum of achievable rates of computing the network's target function.

One fundamental objective in the present paper is to understand the performance of linear network codes for computing different types of target functions. Specifically, we compare the linear computing capacity with that of the (nonlinear) computing capacity and the routing computing capacity for various different classes of target functions in single-receiver networks. Such classes include reducible, injective, semi-injective, and linear target functions over finite fields. Informally, a target function is semi-injective if it uniquely maps at least one of its inputs, and a target function is reducible if it can be computed using a linear transformation followed by a function whose domain has a reduced dimension. Computing capacity bounds and achievability are given with respect to the target function classes studied for network codes that use routing, linear coding, or nonlinear coding.

Our specific contributions will be summarized next.

1.1   Contributions

Section 2 gives many of the formal definitions used in the paper (e.g. target function classes and computing capacity types). We show that routing messages through the intermediate nodes in a network forces the receiver to obtain all the messages even though only a function of the messages is required (Theorem 2.10), and we bound the computing capacity gain of using nonlinear versus routing codes (Theorem 2.12).

In Section 3, we demonstrate that the performance of optimal linear codes may depend on how 'linearity' is defined (Theorem 3.2). Specifically, we show that the linear computing capacity of a network varies depending on which ring linearity is defined over on the source alphabet.

In Sections 4 and 5, we study the computing capacity gain of using linear coding over routing, and nonlinear coding over linear coding. In particular, we study various classes of target functions, including injective, semi-injective, reducible, and linear. The relationships between these classes is illustrated in Figure 1.

Section 4 studies linear coding for network computing. We show that if a target function is not reducible, then the linear computing capacity and routing computing capacity are equal whenever the source alphabet is a finite field (Theorem 4.8); the same result also holds for semi-injective target functions over rings. We also show that whenever a target function is injective, routing obtains the full computing capacity of a network (Theorem 4.9), although whenever a target function is neither reducible nor injective, there exists a network such that the computing capacity is larger than the linear computing capacity (Theorem 4.11). Thus for non-injective target functions that are not reducible, any computing capacity gain of using coding over routing must be obtained through nonlinear coding. This result is tight in the sense that if a target function is reducible, then there always exists a network where the linear computing capacity is larger than the routing capacity (Theorem 4.12). We also show that there exists a reducible target function and a network whose computing capacity is strictly greater than its linear computing capacity, which in turn is strictly greater than its routing computing capacity. (Theorem 4.14).

Section 5 focuses on computing linear target functions over finite fields. We characterize the linear computing capacity for linear target functions over finite fields in arbitrary networks (Theorem 5.6). We show that linear codes are sufficient for linear target functions and we upper bound the computing capacity gain of coding (linear or nonlinear) over routing (Theorem 5.7). This upper bound is shown to be achievable for every linear target function and an associated network, in which case the computing capacity is equal to the routing computing capacity times the number of network sources (Theorem 5.8).

Finally, Section 6 studies an illustrative example for the computing problem, namely the reverse butterfly network – obtained by reversing the direction of all the edges in the multicast butterfly network (the butterfly network studied in [1] illustrated the capacity gain of network coding over routing). For this network and the arithmetic sum target function, we evaluate the routing and linear computing capacity (Theorem 6.1) and the computing capacity (Theorem 6.3). We show that the latter is strictly larger than the first two, which are equal to each other. No network with such properties is presently known for network coding. Among other things, the reverse butterfly network also illustrates that the computing capacity can be a function of the coding alphabet (i.e. the domain of the target function f). In contrast, for network coding, the coding capacity and

routing capacity are known to be independent of the coding alphabet used [5].

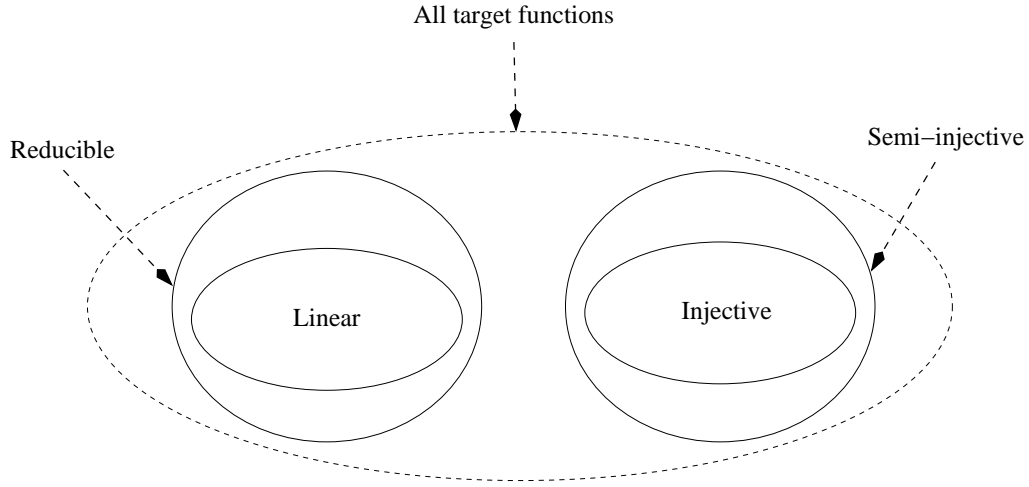Our main results are summarized in Table 1.

Figure 1: Decomposition of the space of all target functions into various classes.

Result	f	$\mathcal{A}$	Location
$\forall f \; \forall \mathcal{N} \; \mathcal{C}_{\text{lin}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	non-reducible	field	Theorem 4.8
	semi-injective	ring	
$\forall f \; \forall \mathcal{N} \; \mathcal{C}_{\text{cod}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	injective		Theorem 4.9
$\forall f \; \exists \mathcal{N} \; \mathcal{C}_{\text{cod}}(\mathcal{N}, f) > \mathcal{C}_{\text{lin}}(\mathcal{N}, f)$	non-injective & non-reducible	field	Theorem 4.11
$\forall f \; \exists \mathcal{N} \; \mathcal{C}_{\text{lin}}(\mathcal{N}, f) > \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	reducible	ring	Theorem 4.12
$\exists f \; \exists \mathcal{N} \; \mathcal{C}_{\text{cod}}(\mathcal{N}, f) > \mathcal{C}_{\text{lin}}(\mathcal{N}, f) > \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	reducible		Theorem 4.14
$\forall f \; \forall \mathcal{N} \; \mathcal{C}_{\text{cod}}(\mathcal{N}, f) = \mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq s \, \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	linear	field	Theorem 5.7
$\forall f \; \exists \mathcal{N} \; \mathcal{C}_{\text{lin}}(\mathcal{N}, f) = s \, \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	linear	field	Theorem 5.8
$\exists f \; \exists \mathcal{N} \; \mathcal{C}_{\text{cod}}(\mathcal{N}, f)$ is irrational	arithmetic sum		Theorem 6.3

Table 1: Summary of our main results for certain classes of target functions. The quantities $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$, $\mathcal{C}_{\text{lin}}(\mathcal{N}, f)$, and $\mathcal{C}_{\text{rout}}(\mathcal{N}, f)$ denote the computing capacity, linear computing capacity, and routing computing capacity, respectively, for a network $\mathcal{N}$ with s sources and target function f. The columns labeled f and $\mathcal{A}$ indicate contraints on the target function f and the source alphabet $\mathcal{A}$, respectively.

2   Network model and definitions

In this paper, a *network* $\mathcal{N} = (G, S, \rho)$ consists of a finite, directed acyclic multigraph $G = (\mathcal{V}, \mathcal{E})$, a set $S = \{\sigma_1, \ldots, \sigma_s\} \subseteq \mathcal{V}$ of s distinct *source nodes* and a single *receiver* $\rho \in \mathcal{V}$. We assume that $\rho \notin S$, and that the graph[1] G contains a directed path from every node in $\mathcal{V}$ to the receiver ρ. For each node $u \in \mathcal{V}$, let $\mathcal{E}_{in}(u)$ and $\mathcal{E}_{out}(u)$ denote the in-edges and out-edges of u respectively. We assume (without loss of generality) that if a network node has no in-edges, then it is a source node. If $e = (u, v) \in \mathcal{E}$, we will use the notation *head(e)* $= u$ and *tail(e)* $= v$.

   An *alphabet* is a finite set of size at least two. Throughout this paper, $\mathcal{A}$ will denote a *source alphabet* and $\mathcal{B}$ will denote a *receiver alphabet*. For any positive integer m, any vector $x \in \mathcal{A}^m$, and any $i \in \{1, 2, \ldots, m\}$, let x_i denote the i-th component of x. For any index set $I = \{i_1, i_2, \ldots, i_q\} \subseteq \{1, 2, \ldots, m\}$ with $i_1 < i_2 < \ldots < i_q$, let x_I denote the vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \in \mathcal{A}^{|I|}$. Sometimes we view $\mathcal{A}$ as an algebraic structure such as a ring, i.e., with multiplication and addition. Throughout this paper, vectors will always be taken to be row vectors. Let $\mathbb{F}_q$ denote a finite field of order q. A superscript t will denote the transpose for vectors and matrices.

2.1   Target functions

For a given network $\mathcal{N} = (G, S, \rho)$, we use s throughout the paper to denote the number $|S|$ of receivers in $\mathcal{N}$. For given network $\mathcal{N}$, a *target function* is a mapping

$$f : \mathcal{A}^s \longrightarrow \mathcal{B}.$$

The goal in network computing is to compute f at the receiver ρ, as a function of the source messages. We will assume that all target functions depend on all the network sources (i.e. a target function cannot be a constant function of any one of its arguments). Some example target functions that will be referenced are listed in Table 2.

Target function f	Alphabet $\mathcal{A}$	$f(x_1, \ldots, x_s)$	Comments
identity	arbitrary	$(x_1, \ldots, x_s)$	$\mathcal{B} = \mathcal{A}^s$
arithmetic sum	$\{0, 1, \ldots, q-1\}$	$x_1 + x_2 + \cdots + x_s$	'+' is ordinary integer addition, $\mathcal{B} = \{0, 1, \cdots, s(q-1)\}$
mod r sum	$\{0, 1, \ldots, q-1\}$	$x_1 \oplus x_2 \oplus \ldots \oplus x_s$	$\oplus$ is $\mathrm{mod}\ r$ addition, $\mathcal{B} = \mathcal{A}$
linear	any ring	$a_1 x_1 + a_2 x_2 + \ldots + a_s x_s$	arithmetic in the ring, $\mathcal{B} = \mathcal{A}$
maximum	any ordered set	$\max\{x_1, \ldots, x_s\}$	$\mathcal{B} = \mathcal{A}$

Table 2: Definitions of some target functions.

---

[1]Throughout the remainder of the paper, we use "graph" to mean a multigraph, and in the context of network computing we use "network" to mean a single-receiver network.

Definition 2.1. Let alphabet $\mathcal{A}$ be a ring. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be *reducible* if there exists an integer λ satisfying $\lambda < s$, an $s \times \lambda$ matrix T with elements in $\mathcal{A}$, and a map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that for all $x \in \mathcal{A}^s$,

$$g(xT) = f(x). \tag{1}$$

Reducible target functions are not injective, since, for example, if x and y are distinct elements of the null-space of T, then

$$f(x) = g(xT) = g(0) = g(yT) = f(y).$$

Example 2.2. Suppose the alphabet is $\mathcal{A} = \mathbb{F}_2$ and the target function is

$$f : \mathbb{F}_2^3 \longrightarrow \{0,1\},$$

where

$$f(x) = (x_1 + x_2)x_3.$$

Then, by choosing $\lambda = 2$,

$$T = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and $g(y_1, y_2) = y_1 y_2$, we get

$$\begin{aligned}
g(xT) &= g(x_1 + x_2, x_3) \\
&= (x_1 + x_2)x_3 \\
&= f(x).
\end{aligned}$$

Thus the target function f is reducible.

Example 2.3. The notion of reducibility requires that for a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, the set $\mathcal{A}$ must be a ring. If we impose any ring structure to the domains of the identity, arithmetic sum, maximum, and minimum target functions, then these can be shown (via our Example 4.2 and Lemma 4.3) to be non-reducible.

2.2 Network computing and capacity

Let k and n be positive integers. Given a network $\mathcal{N}$ with source set S and alphabet $\mathcal{A}$, a *message generator* is any mapping

$$\alpha : S \longrightarrow \mathcal{A}^k.$$

For each source $\sigma_i \in S$, $\alpha(\sigma_i)$ is called a *message vector* and its components

$$\alpha(\sigma_i)_1, \ldots, \alpha(\sigma_i)_k$$

are called *messages*[2].

Definition 2.4. A (k, n) *network code in a network* $\mathcal{N}$ consists of the following:

   (i) *Encoding functions* $h^{(e)}$, for every out-edge $e \in \mathcal{E}_{out}(v)$ of every node $v \in \mathcal{V} - \rho$, of the form:

$$h^{(e)} : \left(\prod_{\hat{e} \in \mathcal{E}_{in}(v)} \mathcal{A}^n \right) \times \mathcal{A}^k \longrightarrow \mathcal{A}^n \quad \text{if } v \text{ is a source node}$$

$$h^{(e)} : \prod_{\hat{e} \in \mathcal{E}_{in}(v)} \mathcal{A}^n \longrightarrow \mathcal{A}^n \qquad \text{otherwise.}$$

   (ii) A *decoding function* ψ of the form:

$$\psi : \prod_{\hat{e} \in \mathcal{E}_{in}(v)} \mathcal{A}^n \longrightarrow \mathcal{B}^k.$$

Furthermore, given a (k, n) network code, every edge $e \in \mathcal{E}$ carries a vector z_e of at most n alphabet symbols[3], which is obtained by evaluating the encoding function $h^{(e)}$ on the set of vectors carried by the in-edges to the node and the node's message vector if the node is a source. The objective of the receiver is to compute the target function f of the source messages, for any arbitrary message generator α. More precisely, the receiver constructs a vector of k alphabet symbols, such that for each $i \in \{1, 2, \ldots, k\}$, the i-th component of the receiver's computed vector equals the value of the desired target function f, applied to the i-th components of the source message vectors, for any choice of message generator α.

Definition 2.5. Suppose in a network $\mathcal{N}$, the in-edges of the receiver are $e_1, e_2, \ldots, e_{|\mathcal{E}_{in}(\rho)|}$. A (k, n) network code is said to *compute f in $\mathcal{N}$* if for each $j \in \{1, 2, \ldots, k\}$, and for each message generator α, the decoding function satisfies

$$\psi \left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_{in}(\rho)|}} \right)_j = f \Big((\alpha(\sigma_1)_j, \cdots, \alpha(\sigma_s)_j) \Big). \tag{2}$$

If there exists a (k, n) code that computes f in $\mathcal{N}$, then the rational number k/n is said to be an *achievable computing rate*.

In the network coding literature, one definition of the *coding capacity* of a network is the supremum of all achievable coding rates [5]. We use an analogous definition for the computing capacity.

Definition 2.6. The *computing capacity* of a network $\mathcal{N}$ with respect to a target function f is

$$\mathcal{C}_{\mathrm{cod}}(\mathcal{N}, f) = \sup \left\{ \frac{k}{n} : \exists (k, n) \text{ network code that computes } f \text{ in } \mathcal{N} \right\}.$$

---

[2] For simplicity we assume each source has associated with it exactly one message vector, but all of the results in this paper can readily be extended to the more general case.

[3] By default, we assume that edges carry exactly n symbols.

The notion of linear codes in networks is most often studied with respect to finite fields. Here we will sometimes use more general ring structures.

Definition 2.7. Let alphabet $\mathcal{A}$ be a ring. A (k, n) network code in a network $\mathcal{N}$ is said to be a *linear network code (over $\mathcal{A}$)* if the encoding functions are linear over $\mathcal{A}$.

Definition 2.8. The *linear computing capacity* of a network $\mathcal{N}$ with respect to target function f is

$$\mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f) = \sup \left\{ \frac{k}{n} \ : \ \exists \ (k, n) \text{ linear network code that computes } f \text{ in } \mathcal{N} \right\}.$$

The *routing computing capacity* $\mathcal{C}_{rout}(\mathcal{N}, f)$ is defined similarly by restricting the encoding functions to routing. We call the quantity $\mathcal{C}_{\mathrm{cod}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f)$ the *computing capacity gain* of using nonlinear coding over linear coding. Similar "gains", such as, $\mathcal{C}_{\mathrm{cod}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f)$ and $\mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f)$ are defined.

Note that Definition 2.7 allows linear codes to have nonlinear decoding functions. In fact, since the receiver alphabet $\mathcal{B}$ need not have any algebraic structure to it, linear decoding functions would not make sense in general. We do, however, examine a special case where $\mathcal{B} = \mathcal{A}$ and the target function is linear, in which case we show that linear codes with linear decoders can be just as good as linear codes with nonlinear decoders (Theorem 5.7).

Definition 2.9. A set of edges $C \subseteq \mathcal{E}$ in network $\mathcal{N}$ is said to *separate* sources $\sigma_{m_1}, \ldots, \sigma_{m_d}$ from the receiver ρ, if for each $i \in \{1, 2, \ldots, d\}$, every directed path from σ_{m_i} to ρ contains at least one edge in C. Define

$$I_C = \{i : C \text{ separates } \sigma_i \text{ from the receiver}\}.$$

The set C is said to be a *cut* in $\mathcal{N}$ if it separates at least one source from the receiver (i.e. $|I_C| \geq 1$). We denote by $\Lambda(\mathcal{N})$ the collection of all cuts in $\mathcal{N}$.

Since I_C is the number of sources disconnected by C and there are s sources, we have

$$|I_C| \leq s. \tag{3}$$

For network coding with a single receiver node and multiple sources (where the receiver demands all the source messages), routing is known to be optimal [23]. Let $\mathcal{C}_{\mathrm{rout}}(\mathcal{N})$ denote the routing capacity of the network $\mathcal{N}$, or equivalently the routing computing capacity for computing the identity target function. It was observed in [23, Theorem 4.2] that for any single-receiver network $\mathcal{N}$,

$$\mathcal{C}_{\mathrm{rout}}(\mathcal{N}) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|}. \tag{4}$$

The following theorem shows that if the intermediate nodes in a network are restricted to perform routing, then in order to compute a target function the receiver is forced to obtain all the source messages. This fact motivates the use of coding for computing functions in networks.

Theorem 2.10. *If $\mathcal{N}$ is a network with target function f, then*

$$\mathcal{C}_{rout}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}).$$

Proof. Since any routing code that computes the identity target function can be used to compute any target function f, we have

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) \geq \mathcal{C}_{\text{rout}}(\mathcal{N}).$$

Conversely, it is easy to see that every component of every source message must be received by ρ in order to compute f, so

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) \leq \mathcal{C}_{\text{rout}}(\mathcal{N}).$$

∎

Theorem 2.12 below gives a general upper bound on how much larger the computing capacity can be relative to the routing computing capacity. It will be shown later, in Theorem 5.7, that for linear target functions over finite fields, the bound in Theorem 2.12 can be tightened by removing the logarithm term.

Lemma 2.11. *If $\mathcal{N}$ is network with a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq (\log_2 |\mathcal{A}|) \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Proof. Using [4, Theorem II.1], one finds the term min-cut$(\mathcal{N}, f)$ defined in [4, Equation (3)] in terms of a quantity $R_{I_C, f}$, which in turn is defined in [4, Definition 1.5]. Since target functions are restricted to not being constant functions of any of their arguments, we have $R_{I_C, f} \geq 2$, from which the result follows. ∎

Theorem 2.12. *If $\mathcal{N}$ is network with a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq s \, (\log_2 |\mathcal{A}|) \, \mathcal{C}_{rout}(\mathcal{N}, f)$$

Proof.

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \leq (\log_2 |\mathcal{A}|) \min_{C \in \Lambda(\mathcal{N})} |C| \qquad \text{[from Lemma 2.11]}$$

$$\leq s \, (\log_2 |\mathcal{A}|) \, \mathcal{C}_{\text{rout}}(\mathcal{N}, f). \qquad \text{[from (3), (4), and Theorem 2.10]}$$

∎

3   Linear coding over different ring alphabets

Whereas the size of a finite field characterizes the field, there are, in general, different rings of the same size, so one must address whether the linear computing capacity of a network might depend on which ring is chosen for the alphabet. In this section, we illustrate this possibility with a specific computing problem.

Let $\mathcal{A} = \{a_0, a_1, a_2, a_3\}$ and let $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ be as defined in Table 3. We consider

f	a_0	a_1	a_2	a_3
a_0	0	1	1	2
a_1	1	0	2	1
a_2	1	2	0	1
a_3	2	1	1	0

Table 3: Definition of the 4-ary map f.

different rings R of size 4 for $\mathcal{A}$ and evaluate the linear computing capacity of the network $\mathcal{N}_1$ shown in Figure 2 with respect to the target function f. Specifically, we let R be either the ring $\mathbb{Z}_4$ of integers modulo 4 or the product ring $\mathbb{Z}_2 \times \mathbb{Z}_2$ of 2-dimensional binary vectors. Denote the linear computing capacity here by

$$\mathcal{C}_{\text{lin}}(\mathcal{N}_1)^R = \sup \left\{ \frac{k}{n} \; : \; \exists \, (k, n) \; R\text{-linear code that computes } f \text{ in } \mathcal{N} \right\}.$$

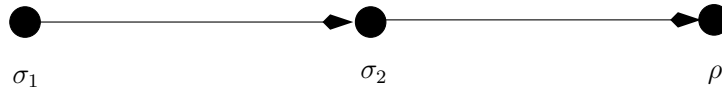The received vector z at ρ can be viewed as a function of the source vectors generated at σ_1 and

Figure 2: Network $\mathcal{N}_1$ has two sources σ_1 and σ_2 and a receiver ρ.

σ_2. For any (k, n) R-linear code, there exist $k \times n$ matrices M_1 and M_2 such that z can be written as

$$z(\alpha(\sigma_1), \alpha(\sigma_2)) = \alpha(\sigma_1) M_1 + \alpha(\sigma_2) M_2. \tag{5}$$

Let $m_{i,1}, \cdots, m_{i,k}$ denote the row vectors of M_i, for $i \in \{1, 2\}$.

Lemma 3.1. *Let $\mathcal{A}$ be the ring $\mathbb{Z}_4$ and let $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ be the target function shown in Table 3, where $a_i = i$, for each i. If a (k, n) linear code over $\mathcal{A}$ computes f in $\mathcal{N}_1$ and ρ receives a zero vector, then $\alpha(\sigma_1) = \alpha(\sigma_2) \in \{0, 2\}^k$.*

Proof. If $\alpha(\sigma_1) = \alpha(\sigma_2) = 0$, then ρ receives a 0 by (5) and must decode a 0 since $f((0,0)) = 0$ (from Table 3). Thus, ρ always decodes a 0 upon receiving a 0. But $f((x_1, x_2)) = 0$ if and only if $x_1 = x_2$ (from Table 3), so whenever ρ receives a 0, the source messages satisfy $\alpha(\sigma_1) = \alpha(\sigma_2)$.

Now suppose, contrary to the lemma's assertion, that there exist messages $\alpha(\sigma_1)$ and $\alpha(\sigma_2)$ such that $z(\alpha(\sigma_1), \alpha(\sigma_2)) = 0$ and $\alpha(\sigma_1)_j \notin \{0, 2\}$ for some $j \in \{1, 2, \cdots, k\}$. Since $\alpha(\sigma_1)_j$ is invertible in $\mathbb{Z}_4$ (it is either 1 or 3), we have from (5) that

$$m_{1,j} = \sum_{\substack{i=1 \\ i \neq j}}^{k} -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_1)_i \, m_{1,i} + \sum_{i=1}^{k} -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_2)_i \, m_{2,i} \tag{6}$$

$$= y^{(1)} M_1 + y^{(2)} M_2 \tag{7}$$

where $y^{(1)}$ and $y^{(2)}$ are k-dimensional vectors defined by

$$y_i^{(1)} = \begin{cases} -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_1)_i & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

$$y_i^{(2)} = -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_2)_i. \tag{8}$$

Also, define the k-dimensional vector x by

$$x_i = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \tag{9}$$

We have from (5) that $z(x, 0) = m_{1,j}$ and from (5) and (7) that $z(y^{(1)}, y^{(2)}) = m_{1,j}$. Thus, in order for the code to compute f, we must have $f(x_j, 0) = f(y_j^{(1)}, y_j^{(2)})$. But $f(x_j, 0) = f(1, 0) = 1$ and

$$\begin{aligned} f(y_j^{(1)}, y_j^{(2)}) &= f(0, -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_2)_j) \\ &= f(0, -\alpha(\sigma_1)_j^{-1} \alpha(\sigma_1)_j) && [\text{from } \alpha(\sigma_1) = \alpha(\sigma_2)] \\ &= f(0, -1) \\ &= f(0, 3) && [\text{from } 3 = -1 \text{ in } \mathbb{Z}_4] \\ &= 2 && [\text{from Table 3}], \end{aligned}$$

a contradiction. Thus, $\alpha(\sigma_1) \in \{0, 2\}^k$. ∎

Theorem 3.2. *The network $\mathcal{N}_1$ in Figure 2 with alphabet $\mathcal{A} = \{a_0, a_1, a_2, a_3\}$ and target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ shown in Table 3, satisfies*

$$\mathcal{C}_{lin}(\mathcal{N}_1, f)^{\mathbb{Z}_4} \leq \frac{2}{3}$$
$$\mathcal{C}_{lin}(\mathcal{N}_1, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} = 1.$$

(For $\mathcal{A} = \mathbb{Z}_4$, we identify $a_i = i$, for each i, and for $\mathcal{A} = \mathbb{Z}_2 \times \mathbb{Z}_2$, we identify each a_i with the

2-bit binary representation of i.)

Proof. Consider a (k, n) $\mathbb{Z}_2 \times \mathbb{Z}_2$-linear code that computes f. From (5), we have $z(x, 0) = 0$ whenever $x M_1 = 0$. Since $f((0, 0)) \neq f((x_i, 0))$ (whenever $x_i \neq 0$), it must therefore be the case that $x M_1 = 0$ only when $x = 0$, or in other words, the rows of M_1 must be independent, so $n \geq k$. Thus,

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} \leq 1. \tag{10}$$

Now suppose that $\mathcal{A}$ is the ring $\mathbb{Z}_2 \times \mathbb{Z}_2$ where, $a_0 = (0, 0)$, $a_1 = (0, 1)$, $a_2 = (1, 0)$, and $a_3 = (1, 1)$ and let $\oplus$ denote the addition over $\mathcal{A}$. For any $x \in \mathcal{A}^2$, the value $f(x)$, as defined in Table 3, is seen to be the Hamming distance between x_1 and x_2. If $k = n = 1$ and $M_1 = M_2 = [a_3]$ (i.e., the 1×1 identity matrix), then ρ receives $x_1 \oplus x_2$ from which f can be computed by summing its components. Thus, a computing rate of $k/n = 1$ is achievable. From (10), it then follows that

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} = 1.$$

We now prove that $\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_4} \leq 2/3$. Let $\mathcal{A}$ denote the ring $\mathbb{Z}_4$ where $a_i = i$ for $0 \leq i \leq 3$. For a given (k, n) linear code over $\mathcal{A}$ that computes f, the n-dimensional vector received by ρ can be written as in (5). Let $\mathcal{K}$ denote the collection of all message vector pairs $(\alpha(\sigma_1), \alpha(\sigma_2))$ such that $z(\alpha(\sigma_1), \alpha(\sigma_2)) = 0$. Define the $2k \times n$ matrix

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

and notice that $\mathcal{K} = \{y \in \mathcal{A}^{2k} : yM = 0\}$. Then,

$$
\begin{aligned}
4^n &= |\mathcal{A}|^n \\
&\geq \left| \{yM : y \in \mathcal{A}^{2k}\} \right| \quad && [\text{from } y \in \mathcal{A}^{2k} \implies yM \in \mathcal{A}^n] \\
&\geq \frac{|\mathcal{A}|^{2k}}{|\mathcal{K}|} \quad && [\text{from } y^{(1)}, y^{(2)} \in \mathcal{A}^{2k} \text{ and } y^{(1)}M = y^{(2)}M \implies y^{(1)} - y^{(2)} \in \mathcal{K}] \\
&\geq \frac{|\mathcal{A}|^{2k}}{2^k} \quad && [\text{from Lemma 3.1}] \\
&= 4^{3k/2}. \quad && [\text{from } |\mathcal{A}| = 4]
\end{aligned}
$$

Thus, $k/n \leq 2/3$, so $\mathcal{C}_{\text{lin}}(\mathcal{N}_1, f)^{\mathbb{Z}_4} \leq \frac{2}{3}$. ∎

4   Linear network codes for computing target functions

Theorem 2.10 showed that if intermediate network nodes use routing, then a network's receiver learns all the source messages irrespective of the target function it demands. In Section 4.1, we prove a similar result when the intermediate nodes use linear network coding. It is shown that whenever a target function is not reducible the linear computing capacity coincides with the routing capacity and the receiver must learn all the source messages. We also show that there exists a network such that the computing capacity is larger than the routing capacity whenever the target function is non-injective. Hence, if the target function is not reducible, such capacity gain must be obtained from nonlinear coding. Section 4.2 shows that linear codes may provide a computing capacity gain over routing for reducible target functions and that linear codes may not suffice to obtain the full computing capacity gain over routing.

4.1   Non-reducible target functions

Verifying whether or not a given target function is reducible may not be easy. We now define a class of target functions that are easily shown to not be reducible.

Definition 4.1. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be *semi-injective* if there exists $x \in \mathcal{A}^s$ such that $f^{-1}(\{f(x)\}) = \{x\}$.

Note that injective functions are semi-injective.

Example 4.2. If f is the arithmetic sum target function, then f is semi-injective (since $f(x) = 0$ implies $x = 0$) but not injective (since $f(0,1) = f(1,0) = 1$). Other examples of semi-injective target functions include the identity, maximum, and minimum functions.

Lemma 4.3. *If alphabet $\mathcal{A}$ is a ring, then semi-injective target functions are not reducible.*

Proof. Suppose that a target function f is reducible. Then there exists an integer λ satisfying $\lambda < s$, matrix $T \in \mathcal{A}^{s \times \lambda}$, and map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that

$$g(xT) = f(x) \ \text{ for each } x \in \mathcal{A}^s. \tag{11}$$

Since $\lambda < s$, there exists a non-zero $d \in \mathcal{A}^s$ such that $dT = 0$. Then for each $x \in \mathcal{A}^s$,

$$f(d + x) = g((d + x)T) = g(xT) = f(x) \tag{12}$$

so f is not semi-injective. $\blacksquare$

Definition 4.4. Let $\mathcal{A}$ be a finite field and let $\mathcal{M}$ be a subspace of the vector space $\mathcal{A}^s$ over the scalar field $\mathcal{A}$. Let

$$\mathcal{M}^\perp = \left\{ y \in \mathcal{A}^s : xy^t = 0 \text{ for all } x \in \mathcal{M} \right\}$$

and let $\dim(\mathcal{M})$ denote the dimension of $\mathcal{M}$ over $\mathcal{A}$.

Lemma 4.5. [4] *If $\mathcal{A}$ is a finite field and $\mathcal{M}$ is a subspace of vector space $\mathcal{A}^s$, then $(\mathcal{M}^\perp)^\perp = \mathcal{M}$.*

Lemma 4.6 will be used in Theorem 4.8. The lemma states an alternative characterization of reducible target functions when the source alphabet is a finite field and of semi-injective target functions when the source alphabet is a group.

Lemma 4.6. *Let $\mathcal{N}$ be a network with target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and alphabet $\mathcal{A}$.*

(i) *Let $\mathcal{A}$ be a finite field. f is reducible if and only if there exists a non-zero $d \in \mathcal{A}^s$ such that for each $a \in \mathcal{A}$ and each $x \in \mathcal{A}^s$,*

$$f(ad + x) = f(x).$$

(ii) *Let $\mathcal{A}$ be a group. f is semi-injective if and only if there exists $x \in \mathcal{A}^s$ such that for every non-zero $d \in \mathcal{A}^s$,*

$$f(d + x) \neq f(x).$$

(The arithmetic in $ad + x$ and $d + x$ is performed component-wise over the corresponding $\mathcal{A}$.)

Proof. (i) If f is reducible, then there exists an integer λ satisfying $\lambda < s$, matrix $T \in \mathcal{A}^{s \times \lambda}$, and map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that

$$g(xT) = f(x) \text{ for each } x \in \mathcal{A}^s. \tag{13}$$

Since $\lambda < s$, there exists a non-zero $d \in \mathcal{A}^s$ such that $dT = 0$. Then for each $a \in \mathcal{A}$ and each $x \in \mathcal{A}^s$,

$$f(ad + x) = g((ad + x)T) = g(xT) = f(x). \tag{14}$$

Conversely, suppose that there exists a non-zero d such that (14) holds for every $a \in \mathcal{A}$ and every $x \in \mathcal{A}^s$ and let $\mathcal{M}$ be the one-dimensional subspace of $\mathcal{A}^s$ spanned by d. Then

$$f(t + x) = f(x) \text{ for every } t \in \mathcal{M}, x \in \mathcal{A}^s. \tag{15}$$

Note that $\dim(\mathcal{M}^\perp) = s - 1$. Let $\lambda = s - 1$, let $T \in \mathcal{A}^{s \times \lambda}$ be a matrix such that its columns form a basis for $\mathcal{M}^\perp$, and let $\mathcal{R}_T$ denote the row space of T. Define the map

$$g : \mathcal{R}_T \longrightarrow f(\mathcal{A}^s)$$

as follows. For any $y \in \mathcal{R}_T$ such that $y = xT$ for $x \in \mathcal{A}^s$, let

$$g(y) = g(xT) = f(x). \tag{16}$$

---

[4] This lemma is a standard result in coding theory regarding dual codes over finite fields, even though the operation xy^t is not an inner product (e.g. [12, Theorem 7.5] or [18, Corollary 3.2.3]). An analogous result for orthogonal complements over inner product spaces is well known in linear algebra (e.g. [13, Theorem 5 on pg. 286]).

Note that if $y = x^{(1)}T = x^{(2)}T$ for $x^{(1)} \neq x^{(2)}$, then

$$
\begin{aligned}
(x^{(1)} - x^{(2)})T &= 0 \\
x^{(1)} - x^{(2)} &\in (\mathcal{M}^\perp)^\perp & \text{[from construction of } T] \\
x^{(1)} - x^{(2)} &\in \mathcal{M} & \text{[from Lemma 4.5]} \\
f(x^{(1)}) &= f((x^{(1)} - x^{(2)}) + x^{(2)}) \\
&= f(x^{(2)}). & \text{[from (15)]}
\end{aligned}
$$

Thus g is well defined. Then from (16) and Definition 2.1, f is reducible.

(ii) Since f is semi-injective, there exists a $x \in \mathcal{A}^s$ such that $\{x\} = f^{-1}(\{f(x)\})$, which in turn is true if and only if for each non-zero $d \in \mathcal{A}^s$, we have $f(d + x) \neq f(x)$.

∎

The following example shows that if the alphabet $\mathcal{A}$ is not a finite field, then the assertion in Lemma 4.6(i) may not be true.

Example 4.7. Let $\mathcal{A} = \mathbb{Z}_4$, let $f : \mathcal{A} \longrightarrow \mathcal{A}$ be the target function defined by $f(x) = 2x$, and let $d = 2$. Then, for all $a \in \mathcal{A}$,

$$
\begin{aligned}
f(2a + x) &= 2(2a + x) \\
&= 2x & \text{[from } 4 = 0 \text{ in } \mathbb{Z}_4] \\
&= f(x)
\end{aligned}
$$

but, f is not reducible, since $s = 1$.

Theorem 4.8 establishes for a network with a finite field alphabet, whenever the target function is not reducible, linear computing capacity is equal to the routing computing capacity, and therefore if a linear network code is used, the receiver ends up learning all the source messages even though it only demands a function of these messages.

For network coding (i.e. when f is the identity function), many multi-receiver networks have a larger linear capacity than their routing capacity. However, all single-receiver networks are known to achieve their coding capacity with routing [23]. For network computing, the next theorem shows that with non-reducible target functions there is no advantage to using linear coding over routing.[5]

Theorem 4.8. *Let $\mathcal{N}$ be a network with target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and alphabet $\mathcal{A}$. If $\mathcal{A}$ is a finite field and f is not reducible, or $\mathcal{A}$ is a ring with identity and f is semi-injective, then*

$$
\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f).
$$

Proof. Since any routing code is in particular a linear code,

$$
\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \geq \mathcal{C}_{\text{rout}}(\mathcal{N}, f).
$$

---

[5] As a reminder, "network" here refers to single-receiver networks in the context of computing.

Now consider a (k, n) linear code that computes the target function f in $\mathcal{N}$ and let C be a cut. We will show that for any two collections of source messages, if the messages agree at sources not separated from ρ by C and the vectors agree on edges in C, then there exist two other source message collections with different target function values, such that the receiver ρ cannot distinguish this difference. In other words, the receiver cannot properly compute the target function in the network.

For each $e \in C$, there exist $k \times n$ matrices $M(e)_1, \ldots, M(e)_s$ such that the vector carried on e is

$$\sum_{i=1}^{s} \alpha(\sigma_i) \, M(e)_i.$$

For any matrix M, denote its j-th column by $M^{(j)}$. Let w and y be different $k \times s$ matrices over $\mathcal{A}$, whose j-th columns agree for all $j \notin I_C$.

Let us suppose that the vectors carried on the edges of C, when the the column vectors of w are the source messages, are the same as when the the column vectors of y are the source messages. Then, for all $e \in C$,

$$\sum_{i=1}^{s} w^{(i)} M(e)_i = \sum_{i=1}^{s} y^{(i)} M(e)_i. \tag{17}$$

We will show that this leads to a contradiction, namely that ρ cannot compute f. Let m be an integer such that if d denotes the m-th row of $w - y$, then $d \neq 0$. For the case where $\mathcal{A}$ is a field and f is not reducible, by Lemma 4.6(i), there exist $a \in \mathcal{A}$ and $x \in \mathcal{A}^s$ such that $ad \neq 0$ and

$$f(ad + x) \neq f(x). \tag{18}$$

In the case where $\mathcal{A}$ is a ring with identity and f is semi-injective, we obtain (18) from Lemma 4.6(ii) in the special case of $a = 1$.

Let u be any $k \times s$ matrix over $\mathcal{A}$ whose m-th row is x and let $v = u + a(w - y)$. From (18), the target function f differs on the m-th rows of u and v. Thus, the vectors on the in-edges of the receiver ρ must differ between two cases: (1) when the sources messages are the columns of u, and (2) when the sources messages are the columns of v. The vector carried by any in-edge of the receiver is a function of each of the message vectors $\alpha(\sigma_j)$, for $j \notin I_C$, and the vectors carried by the edges in the cut C. Furthermore, the j-th columns of u and v agree if $j \notin I_C$. Thus, at least one of the vectors on an edge in C must change when the set of source message vectors changes from u to v. However this is contradicted by the fact that for all $e \in C$, the vector carried on e when the columns of u are the source messages is

$$\sum_{i=1}^{s} u^{(i)} M(e)_i = \sum_{i=1}^{s} u^{(i)} M(e)_i + a \sum_{i=1}^{s} (w^{(i)} - y^{(i)})) M(e)_i \qquad \text{[from (17)]}$$

$$= \sum_{i=1}^{s} v^{(i)} M(e)_i \tag{19}$$

which is also the vector carried on e when the columns of v are the source messages.

Hence, for any two different matrices w and y whose j-th columns agree for all $j \notin I_C$, at least one vector carried by an edge in the cut C has to differ in value in the case where the source messages are the columns of w from the case where the source messages are the columns of y. This fact implies that

$$(|\mathcal{A}|^n)^{|C|} \geq (|\mathcal{A}|^k)^{|I_C|}$$

and thus

$$\frac{k}{n} \leq \frac{|C|}{|I_C|}.$$

Since the cut C is arbitrary, we conclude (using (4)) that

$$\frac{k}{n} \leq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|} = \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f).$$

Taking the supremum over all (k, n) linear network codes that compute f in $\mathcal{N}$, we get

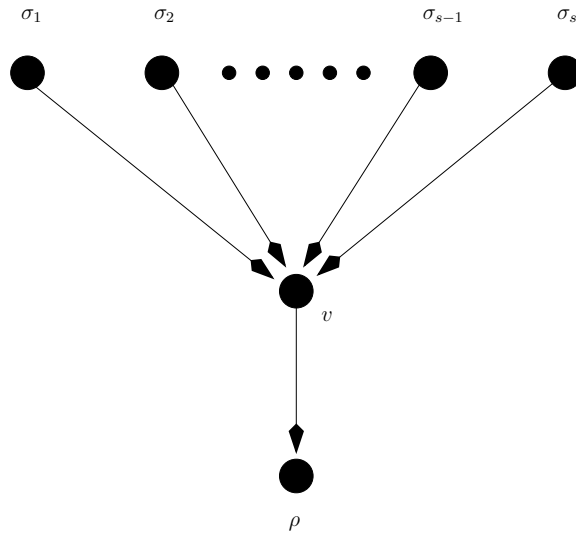$$\mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f) \leq \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f).$$

■

Figure 3: Network $\mathcal{N}_{2,s}$ has sources $\sigma_1, \sigma_2, \ldots, \sigma_s$, each connected to the relay v by an edge and v is connected to the receiver by an edge.

Theorem 4.8 showed that if a network's target function is not reducible (e.g. semi-injective target functions) then there can be no computing capacity gain of using linear coding over routing. The following theorem shows that if the target function is injective, then there cannot even be any nonlinear computing gain over routing.

Note that if the identity target function is used in Theorem 4.9, then the result states that there is no coding gain over routing for ordinary network coding. This is consistent since our stated assumption in Section 2 is that only single-receiver networks are considered here (for some networks with two or more receivers, it is well known that linear coding may provide network coding gain over network routing).

Theorem 4.9. *If $\mathcal{N}$ is a network with an injective target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f).$$

Proof. It follows from [23, Theorem 4.2] that for any single-receiver network $\mathcal{N}$ and the identity target function f, we have $\mathcal{C}_{cod}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f)$. This can be straightforwardly extended to injective target functions for network computing. ∎

Theorem 4.8 showed that there cannot be linear computing gain for networks whose target functions are not reducible, and Theorem 4.9 showed that the same is true for target functions that are injective. However, Theorem 4.11 will show via an example network that nonlinear codes may provide a capacity gain over linear codes if the target function is not injective. This reveals a limitation of linear codes compared to nonlinear ones for non-injective target functions that are not reducible. For simplicity, in Theorem 4.11 we only consider the case when there are two or more sources. We need the following lemma first.

Lemma 4.10. *The computing capacity of the network $\mathcal{N}_{2,s}$ shown in Figure 3, with respect to a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, satisfies*

$$\mathcal{C}_{cod}(\mathcal{N}_{2,s}, f) \geq \min\left\{1, \ \frac{1}{\log_{|\mathcal{A}|} |f\left(\mathcal{A}^s\right)|}\right\}.$$

Proof. Suppose

$$\log_{|\mathcal{A}|} |f\left(\mathcal{A}^s\right)| < 1. \tag{20}$$

Let $k = n = 1$ and assume that each source node sends its message to node v. Let

$$g \ : \ f\left(\mathcal{A}^s\right) \longrightarrow \mathcal{A}$$

be any injective map (which exists by (20)). Then the node v can compute g and send it to the receiver. The receiver can compute the value of f from the value of g and thus a rate of 1 is achievable, so $\mathcal{C}_{cod}(\mathcal{N}_{2,s}, f) \geq 1$.

Now suppose

$$\log_{|\mathcal{A}|} |f\left(\mathcal{A}^s\right)| \geq 1. \tag{21}$$

Choose integers k and n such that

$$\frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} - \epsilon \leq \frac{k}{n} \leq \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|}. \tag{22}$$

Now choose an arbitrary injective map (which exists by (22))

$$g \ : \ (f(\mathcal{A}^s))^k \longrightarrow \mathcal{A}^n.$$

Since $n \geq k$ (by (21) and (22)), we can still assume that each source sends its k-length message vector to node v. Node v computes f for each of the k sets of source messages, encodes those values into an n-length vector over $\mathcal{A}$ using the injective map g and transmits it to the receiver. The existence of a decoding function which satisfies (2) is then obvious from the fact that g is injective. From (22), the above code achieves a computing rate of

$$\frac{k}{n} \geq \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} - \epsilon.$$

Since ϵ was arbitrary, it follows that the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_{2,s}, f)$ is at least $1/\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|$. ∎

Theorem 4.11. *Let $\mathcal{A}$ be a finite field alphabet. Let $s \geq 2$ and let f be a target function that is neither injective nor reducible. Then there exists a network $\mathcal{N}$ such that*

$$\mathcal{C}_{cod}(\mathcal{N}, f) > \mathcal{C}_{lin}(\mathcal{N}, f).$$

Proof. If $\mathcal{N}$ is the network $\mathcal{N}_{2,s}$ shown in Figure 3 with alphabet $\mathcal{A}$, then

$$\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}, f) &= 1/s & &\text{[from Theorem 4.8 and (4)]} \\
&< \min\left\{1, \ \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|}\right\} & &\text{[from } s \geq 2 \text{ and } |f(\mathcal{A}^s)| < |\mathcal{A}|^s\text{]} \\
&\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f). & &\text{[from Lemma 4.10]}
\end{aligned}$$

∎

The same proof of Theorem 4.11 shows that it also holds if the alphabet $\mathcal{A}$ is a ring with identity and the target function f is semi-injective but not injective.

4.2 Reducible target functions

In Theorem 4.12, we prove a converse to Theorem 4.8 by showing that if a target function is reducible, then there exists a network in which the linear computing capacity is larger than the routing computing capacity. Theorem 4.14 shows that, even if the target function is reducible, linear codes may not achieve the full (nonlinear) computing capacity of a network.

Theorem 4.12. *Let $\mathcal{A}$ be a ring. If a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is reducible, then there exists a network $\mathcal{N}$ such that*

$$\mathcal{C}_{lin}(\mathcal{N}, f) > \mathcal{C}_{rout}(\mathcal{N}, f).$$

Proof. Since f is reducible, there exist $\lambda < s$, a matrix $T \in \mathcal{A}^{s \times \lambda}$, and a map $g : \mathcal{A}^\lambda \longrightarrow f(\mathcal{A}^s)$ such that

$$g(xT) = f(x) \text{ for every } x \in \mathcal{A}^s. \qquad \text{[from Definition 2.1]} \qquad (23)$$

Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,s}$ with alphabet $\mathcal{A}$ and target function f. Let $k = 1$, $n = \lambda$ and let the decoding function be $\psi = g$. Since $n \geq 1$, we assume that all the source nodes transmit their messages to node v. For each source vector

$$x = (\alpha(\sigma_1), \alpha(\sigma_2), \ldots, \alpha(\sigma_s))$$

node v computes xT and sends it to the receiver. Having received the n-dimensional vector xT, the receiver computes

$$
\begin{aligned}
\psi(xT) &= g(xT) & \text{[from } \psi = g] \\
&= f(x). & \text{[from (23)]}
\end{aligned}
$$

Thus there exists a linear code that computes f in $\mathcal{N}$ with an achievable computing rate of

$$
\begin{aligned}
\frac{k}{n} &= \frac{1}{\lambda} \\
&> 1/s & \text{[from } \lambda \leq s - 1] \\
&= \mathcal{C}_{rout}(\mathcal{N}) & \text{[from (4)]}
\end{aligned}
$$

which is sufficient to establish the claim. ∎

For target functions that are not reducible, any improvement on achievable rate of computing using coding must be provided by nonlinear codes (by Theorem 4.8). However, within the class of reducible target functions, it turns out that there are target functions for which linear codes are optimal (i.e., capacity achieving) as shown in Theorem 5.7, while for certain other reducible target functions, nonlinear codes might provide a strictly larger achievable computing rate compared to linear codes.

Remark 4.13. It is possible for a network $\mathcal{N}$ to have a reducible target function f but satisfy $\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f)$ since the network topology may not allow coding to exploit the structure of the target function to obtain a capacity gain. For example, the 3-node network in Figure 4 with $f(x_1, x_2) = x_1 + x_2$ and finite field alphabet $\mathcal{A}$ has

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f) = 1.$$

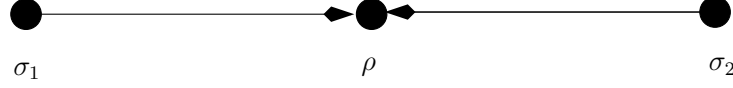$$\sigma_1 \qquad\qquad\qquad \rho \qquad\qquad\qquad \sigma_2$$

Figure 4: A network where there is no benefit to using linear coding over routing for computing f.

Theorem 4.11 shows that for every non-injective, non-reducible target function, some network has a nonlinear computing gain over linear coding, and Theorem 4.12 shows that for every reducible (hence non-injective) target function, some network has a linear computing gain over routing. The following theorem shows that for some reducible target function, some network has both of these linear and nonlinear computing gains.

Theorem 4.14. *There exists a network $\mathcal{N}$ and a reducible target function f such that:*

$$\mathcal{C}_{cod}(\mathcal{N}, f) > \mathcal{C}_{lin}(\mathcal{N}, f) > \mathcal{C}_{rout}(\mathcal{N}, f).$$

Proof. Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,3}$ shown in Figure 3 with $s = 3$, alphabet $\mathcal{A} = \mathbb{F}_2$, and let f be the target function in Example 2.2. The routing capacity is given by

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) = 1/3. \qquad\qquad \text{[from (4)]} \qquad\qquad (24)$$

Let $k = n = 1$. Assume that the sources send their respective messages to node v. The target function f can then be computed at v and sent to the receiver. Hence, $k/n = 1$ is an achievable computing rate and thus

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq 1. \qquad\qquad\qquad\qquad\qquad\qquad (25)$$

Now consider any (k, n) linear code that computes f in $\mathcal{N}$. Such a linear code immediately implies a (k, n) linear code that computes the target function $g(x_1, x_2) = x_1 x_2$ in network $\mathcal{N}_{2,2}$ as follows. From the (k, n) linear code that computes f in $\mathcal{N}$, we get a $3k \times n$ matrix M such that the node v in network $\mathcal{N}$ computes

$$\begin{pmatrix} \alpha(\sigma_1) & \alpha(\sigma_2) & \alpha(\sigma_3) \end{pmatrix} M$$

and the decoding function computes f from the resulting vector. Now, in $\mathcal{N}_{2,2}$, we let the node v compute

$$\begin{pmatrix} \alpha(\sigma_1) & 0 & \alpha(\sigma_2) \end{pmatrix} M$$

and send it to the receiver. The receiver can compute the function g from the received n-dimensional vector using the relation $g(x_1, x_2) = f(x_1, 0, x_2)$. Using the fact that the function g is not reducible (in fact, it is semi-injective),

$$\begin{aligned} \frac{k}{n} &\leq \mathcal{C}_{\text{lin}}(\mathcal{N}_{2,2}, g) \\ &= \mathcal{C}_{\text{rout}}(\mathcal{N}_{2,2}, g) && \text{[from Theorem 4.8]} \\ &= 1/2. && \text{[from (4)]} \end{aligned}$$

Consequently,

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq 1/2. \tag{26}$$

Now we will construct a $(1, 2)$ linear code that computes f in $\mathcal{N}$. Let $k = 1$, $n = 2$ and

$$M = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Let the sources send their respective messages to v while v computes

$$\begin{pmatrix} \alpha(\sigma_1) & \alpha(\sigma_2) & \alpha(\sigma_3) \end{pmatrix} M$$

and transmits the result to the receiver from which f is computable. Since the above code achieves a computing rate of $1/2$, combined with (26), we get

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = 1/2. \tag{27}$$

The claim of the theorem now follows from (24), (25), and (27). ∎

5 Computing linear target functions

We have previously shown that for reducible target functions there may be a computing capacity gain for using linear codes over routing. In this section, we show that for a special subclass of reducible target functions, namely linear target functions[6] over finite fields, linear network codes achieve the full (nonlinear) computing capacity. We now describe a special class of linear codes over finite fields that suffice for computing linear target functions over finite fields at the maximum possible rate.

Throughout this section, let $\mathcal{N}$ be a network and let k, n, and c be positive integers such that $k/n = c$. Each k symbol message vector generated by a source $\sigma \in S$ can be viewed as a c-dimensional vector

$$\alpha(\sigma) = (\alpha(\sigma)_1, \alpha(\sigma)_2, \ldots, \alpha(\sigma)_c) \in \mathbb{F}_{q^k}$$

where $\alpha(\sigma)_i \in \mathbb{F}_{q^n}$ for each i. Likewise, the decoder ψ generates a vector of k symbols from $\mathbb{F}_q$, which can be viewed as a c-dimensional vector of symbols from $\mathbb{F}_{q^n}$. For each $e \in \mathcal{E}$, the edge vector z_e is viewed as an element of $\mathbb{F}_{q^n}$.

For every node $u \in \mathcal{V} - \rho$, and every out-edge $e \in \mathcal{E}_{out}(u)$, we choose an encoding function $h^{(e)}$ whose output is:

$$\begin{cases} \displaystyle\sum_{\hat{e} \in \mathcal{E}_{in}(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}} + \sum_{j=1}^{c} \beta_j^{(e)} \alpha(u)_j & \text{if } u \in S \\ \displaystyle\sum_{\hat{e} \in \mathcal{E}_{in}(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}} & \text{otherwise} \end{cases} \tag{28}$$

for some $\gamma_{\hat{e}}^{(e)}, \beta_j^{(e)} \in \mathbb{F}_{q^n}$ and we use a decoding function ψ whose j-th component output ψ_j is:

$$\sum_{e \in \mathcal{E}_{in}(\rho)} \delta_j^{(e)} z_e \quad \text{for all } j \in \{1, 2, \ldots, c\} \tag{29}$$

for certain $\delta_j^{(e)} \in \mathbb{F}_{q^n}$. Here we view each $h^{(e)}$ as a function of the in-edges to e and the source messages generated by u and we view ψ as a function of the inputs to the receiver. The chosen encoder and decoder are seen to be linear.

Let us denote the edges in $\mathcal{E}$ by $e_1, e_2, \ldots, e_{|\mathcal{E}|}$. For each source σ and each edge $e_j \in \mathcal{E}_{out}(\sigma)$, let $x_1^{(e_j)}, \ldots, x_c^{(e_j)}$ be variables, and for each $e_j \in \mathcal{E}_{in}(\rho)$, let $w_1^{(e_j)}, \ldots, w_c^{(e_j)}$ be variables. For every $e_i, e_j \in \mathcal{E}$ such that $head(e_i) = tail(e_j)$, let $y_{e_i}^{(e_j)}$ be a variable. Let x, y, w be vectors containing all the variables $x_i^{(e_j)}$, $y_{e_i}^{(e_j)}$, and $w_i^{(e_j)}$, respectively. We will use the short hand notation $\mathbb{F}[y]$ to mean the ring of polynomials $\mathbb{F}[\cdots, y_{e_i}^{(e_j)}, \cdots]$ and similarly for $\mathbb{F}[x, y, w]$.

Next, we define matrices $A_\tau(x)$, $F(y)$, and $B(w)$.

---

[6]The definition of "linear target function" was given in Table 2.

(i) For each $\tau \in \{1, 2, \cdots, s\}$, let $A_\tau(x)$ be a $c \times |\mathcal{E}|$ matrix $A_\tau(x)$, given by

$$(A_\tau(x))_{i,j} = \begin{cases} x_i^{(e_j)} & \text{if } e_j \in \mathcal{E}_{out}(\sigma_\tau) \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

(ii) Let $F(y)$ be a $|\mathcal{E}| \times |\mathcal{E}|$ matrix, given by

$$(F(y))_{i,j} = \begin{cases} y_{e_i}^{(e_j)} & \text{if } e_i, e_j \in \mathcal{E} \text{ and } head(e_i) = tail(e_j) \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

(iii) Let $B(w)$ be a $c \times |\mathcal{E}|$ matrix, given by

$$(B(w))_{i,j} = \begin{cases} w_i^{(e_j)} & \text{if } e_j \in \mathcal{E}_{in}(\rho) \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

Consider an (nc, n) linear code of the form in (28)–(29).

Since the graph G associated with the network is acyclic, we can assume that the edges $e_1, e_2, \ldots$ are ordered such that the matrix F is strictly upper-triangular, and thus we can apply Lemma 5.1. Let I denote the identity matrix of suitable dimension.

Lemma 5.1. *(Koetter-Médard [14, Lemma 2]) The matrix $I - F(y)$ is invertible over the ring $\mathbb{F}_q[y]$.*

Lemma 5.2. *(Koetter-Médard [14, Theorem 3]) For $s = 1$ and for all $\tau \in \{1, \ldots, s\}$, the decoder in (29) satisfies*

$$\psi = \alpha(\sigma_1) \, A_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t.$$

Lemma 5.3. *(Alon [2, Theorem 1.2]) Let $\mathbb{F}$ be an arbitrary field, and let $g = g(x_1, \ldots, x_m)$ be a polynomial in $\mathbb{F}[x_1, \ldots, x_m]$. Suppose the degree $deg(g)$ of g is $\sum_{i=1}^m t_i$, where each t_i is a nonnegative integer, and suppose the coefficient of $\prod_{i=1}^m x_i^{t_i}$ in g is nonzero. Then, if $S_1, \ldots, S_m$ are subsets of $\mathbb{F}$ with $|S_i| > t_i$, there are $s_1 \in S_1$, $s_2 \in S_2, \ldots, s_m \in S_m$ so that*

$$g(s_1, \ldots, s_m) \neq 0.$$

For each $\tau \in \{1, 2, \ldots, s\}$, define the $c \times c$ matrix

$$M_\tau(x, y, w) = A_\tau(x)(I - F(y))^{-1} B(w)^t \tag{33}$$

where the components of $M_\tau(x, y, w)$ are viewed as lying in $\mathbb{F}_q[x, y, w]$.

Lemma 5.4. *If for all $\tau \in \{1, 2, \ldots, s\}$,*

$$\det \left(M_\tau(x, y, w) \right) \neq 0$$

in the ring $\mathbb{F}_q[x, y, w]$*, then there exists an integer* $n > 0$ *and vectors* β, γ, δ *over* $\mathbb{F}_{q^n}$ *such that for all* $\tau \in \{1, 2, \ldots, s\}$ *the matrix* $M_\tau(\beta, \gamma, \delta)$ *is invertible in the ring of* $c \times c$ *matrices with components in* $\mathbb{F}_{q^n}$.

Proof. The quantity

$$\det \left(\prod_{\tau=1}^{s} M_\tau(x, y, w) \right)$$

is a nonzero polynomial in $\mathbb{F}_q[x, y, w]$ and therefore also in $\mathbb{F}_{q^n}[x, y, w]$ for any $n \geq 1$. Therefore, we can choose n large enough such that the degree of this polynomial is less than q^n. For such an n, Lemma 5.3 implies there exist vectors β, γ, δ (whose components correspond to the components of the vector variables x, y, w) over $\mathbb{F}_{q^n}$ such that

$$\det \left(\prod_{\tau=1}^{s} M_\tau(\beta, \gamma, \delta) \right) \neq 0. \tag{34}$$

and therefore, for all $\tau \in \{1, 2, \ldots, s\}$

$$\det \left(M_\tau(\beta, \gamma, \delta) \right) \neq 0.$$

Thus, each $M_\tau(\beta, \gamma, \delta)$ is invertible.

∎

The following lemma improves upon the upper bound of Lemma 2.11 in the special case where the target function is linear over a finite field.

Lemma 5.5. *If* $\mathcal{N}$ *is network with a linear target function* f *over a finite field, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Proof. The same argument is used as in the proof of Lemma 2.11, except instead of using $R_{I_C, f} \geq 2$, we use the fact that $R_{I_C, f} = |\mathcal{A}|$ for linear target functions. ∎

Theorem 5.6. *If* $\mathcal{N}$ *is a network with a linear target function* f *over finite field* $\mathbb{F}_q$*, then*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Proof. We have

$$\begin{aligned} \mathcal{C}_{\text{lin}}(\mathcal{N}, f) &\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) \\ &\leq \min_{C \in \Lambda(\mathcal{N})} |C|. \qquad \text{[from Lemma 5.5]} \end{aligned}$$

For a lower bound, we will show that there exists an integer n and an (nc, n) linear code that computes f with a computing rate of $c = \min_{C \in \Lambda(\mathcal{N})} |C|$.

From Lemma 5.1, the matrix $I - F(y)$ in invertible over the ring $\mathbb{F}_q[x, y, w]$ and therefore also over $\mathbb{F}_{q^n}[x, y, w]$. Since any minimum cut between the source σ_τ and the receiver ρ has at least c edges, it follows from [14, Theorem 2][7] that $\det(M_\tau(x, y, w)) \neq 0$ for every $\tau \in \{1, 2, \ldots, s\}$. From Lemma 5.4, there exists an integer $n > 0$ and vectors β, γ, δ over $\mathbb{F}_{q^n}$ such that $M_\tau(\beta, \gamma, \delta)$ is invertible for every $\tau \in \{1, 2, \ldots, s\}$. Since f is linear, we can write

$$f(u_1, \ldots, u_s) = a_1 u_1 + \cdots + a_s u_s.$$

For each $\tau \in \{1, 2, \ldots, s\}$, let

$$\hat{A}_\tau(\beta) = a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta). \tag{35}$$

If a linear code corresponding to the matrices $\hat{A}_\tau(\beta), B(\delta)$, and $F(\gamma)$ is used in network $\mathcal{N}$, then the c-dimensional vector over $\mathbb{F}_{q^n}$ computed by the receiver ρ is

$$
\begin{aligned}
\psi &= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, \hat{A}_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t && \text{[from Lemma 5.2 and linearity]} \\
&= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t && \text{[from (35)]} \\
&= \sum_{\tau=1}^{s} a_\tau\, \alpha(\sigma_\tau) && \text{[from (33)]} \\
&= \left(f(\alpha(\sigma_1)_1, \ldots, \alpha(\sigma_s)_1), \ldots, f(\alpha(\sigma_1)_c, \ldots, \alpha(\sigma_s)_c)\right)
\end{aligned}
$$

which proves that the linear code achieves a computing rate of c. ∎

Theorem 5.7 below proves the optimality of linear codes for computing linear target functions in a single-receiver network. It also shows that the computing capacity of a network for a given target function cannot be larger than the number of network sources times the routing computing capacity for the same target function. This bound tightens the general bound given in Theorem 2.12 for the special case of linear target functions over finite fields. Theorem 5.8 shows that this upper bound can be tight.

Theorem 5.7. *If $\mathcal{N}$ is network with s sources and linear target function f over finite field $\mathbb{F}_q$, then*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{cod}(\mathcal{N}, f) \leq s\, \mathcal{C}_{rout}(\mathcal{N}, f).$$

---

[7]Using the implication (1) $\implies$ (3) in [14, Theorem 2].

Proof.

$$
\begin{aligned}
s\,\mathcal{C}_{\text{rout}}(\mathcal{N}, f) &\geq \min_{C \in \Lambda(\mathcal{N})} |C| && \text{[from (4) and Theorem 2.10]} \\
&\geq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) && \text{[from Lemma 5.5]} \\
&\geq \mathcal{C}_{\text{lin}}(\mathcal{N}, f) \\
&= \min_{C \in \Lambda(\mathcal{N})} |C|. && \text{[from Theorem 5.6]}
\end{aligned}
$$

∎

We note that the inequality in Theorem 5.7 can be shown to apply to certain target functions other than linear functions over finite fields, such as the minimum, maximum, and arithmetic sum target functions.

Theorem 5.8. *For every s, if a target function $f : \mathcal{A}^s \longrightarrow \mathcal{A}$ is linear over finite field $\mathbb{F}_q$, then there exists a network $\mathcal{N}$ with s sources, such that*

$$
\mathcal{C}_{lin}(\mathcal{N}, f) = s\,\mathcal{C}_{rout}(\mathcal{N}, f).
$$

Proof. Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,s}$ shown in Figure 3. Then

$$
\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}, f) &= 1 && \text{[from Theorem 5.6]} \\
\mathcal{C}_{\text{rout}}(\mathcal{N}, f) &= \mathcal{C}_{\text{rout}}(\mathcal{N}) && \text{[from Theorem 2.10]} \\
&= 1/s. && \text{[from (4)]}
\end{aligned}
$$

∎

6 The reverse butterfly network

In this section we study an example network which illustrates various concepts discussed previously in this paper and also provides some interesting additional results for network computing.
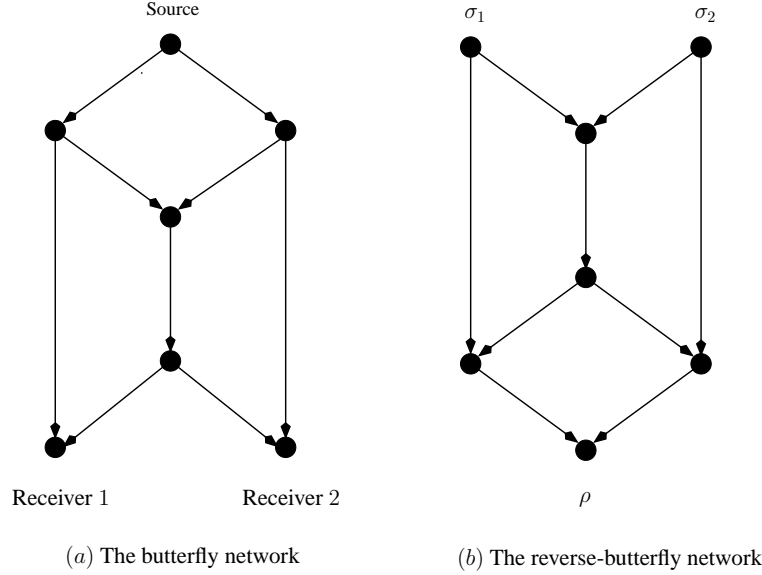

(a) The butterfly network   (b) The reverse-butterfly network

Figure 5: The butterfly network and its reverse $\mathcal{N}_3$.

The network $\mathcal{N}_3$ shown in Figure 5(b) is called the *reverse butterfly network*. It has $S = \{\sigma_1, \sigma_2\}$, receiver node ρ, and is obtained by reversing the direction of all the edges of the multicast butterfly network shown in Figure 5(a).

Theorem 6.1. *The routing and linear computing capacities of the reverse butterfly network $\mathcal{N}_3$ with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and arithmetic sum target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$ are*

$$\mathcal{C}_{rout}(\mathcal{N}_3, f) = \mathcal{C}_{lin}(\mathcal{N}_3, f) = 1.$$

Proof. We have

$$\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}_3, f) &= \mathcal{C}_{\text{rout}}(\mathcal{N}_3) && \text{[from Theorem 4.8]} \\
&= 1. && \text{[from (4)]}
\end{aligned}$$

∎

Remark 6.2. The arithmetic sum target function can be computed in the reverse butterfly network at a computing rate of 1 using only routing (by sending σ_1 down the left side and σ_2 down the right side of the graph). Combined with Theorem 6.1, it follows that the routing computing capacity is equal to 1 for all $q \geq 2$.

Theorem 6.3. *The computing capacity of the reverse butterfly network $\mathcal{N}_3$ with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and arithmetic sum target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$ is*

$$\mathcal{C}_{cod}(\mathcal{N}_3, f) = \frac{2}{\log_q(2q-1)}.$$

Remark 6.4. The computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ obtained in Theorem 6.3 is a function of the coding alphabet $\mathcal{A}$ (i.e. the domain of the target function f). In contrast, for ordinary network coding (i.e. when the target function is the identity map), the coding capacity and routing capacity are known to be independent of the coding alphabet used [5]. For the reverse butterfly network, if, for example, $q = 2$, then $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ is approximately equal to 1.26 and increases asymptotically to 2 as $q \to \infty$.

Remark 6.5. The ratio of the coding capacity to the routing capacity for the multicast butterfly network with two messages was computed in [5] to be $4/3$ (i.e. coding provides a gain of about 33%). The corresponding ratio for the reverse butterfly network increases as a function of q from approximately 1.26 (i.e. 26%) when $q = 2$ to 2 (i.e. 100%) when $q = \infty$. Furthermore, in contrast to the multicast butterfly network, where the coding capacity is equal to the linear coding capacity, in the reverse butterfly network the computing capacity is strictly greater than the linear computing capacity.

Remark 6.6. Recall that capacity is defined as the supremum of a set of rational numbers k/n such that a (k, n) code that computes a target function exists. It was pointed out in [5] that it remains an open question whether the coding capacity of a network can be irrational. Our Theorem 6.3 demonstrates that the computing capacity of a network (e.g. the reverse butterfly network) with unit capacity links can be irrational when the target function to be computed is the arithmetic sum target function of the source messages.

The following lemma is used to prove Theorem 6.3.

Lemma 6.7. *The computing capacity of the reverse butterfly network $\mathcal{N}_3$ with $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and the mod q sum target function f is*

$$\mathcal{C}_{cod}(\mathcal{N}_3, f) = 2.$$

Proof. The upper bound of 2 on $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ follows from [4, Theorem II.1]. To establish the achievability part, let $k = 2$ and $n = 1$. Consider the code shown in Figure 6, where '$\oplus$' indicates the mod q sum. The receiver node ρ gets $\alpha(\sigma_1)_1 \oplus \alpha(\sigma_2)_1$ and $\alpha(\sigma_1)_1 \oplus \alpha(\sigma_2)_1 \oplus \alpha(\sigma_1)_2 \oplus \alpha(\sigma_2)_2$ on its in-edges, from which it can compute $\alpha(\sigma_1)_2 \oplus \alpha(\sigma_2)_2$. This code achieves a rate of 2. ∎

Proof of Theorem 6.3: We have

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \leq 2/\log_q(2q-1). \qquad\qquad \text{[from [4, Theorem II.1]]}$$

To establish the lower bound, we use the fact the that arithmetic sum of two elements from $\mathcal{A} = \{0, 1, \ldots, q-1\}$ is equal to their mod $2q-1$ sum. Let the reverse butterfly network have alphabet $\hat{\mathcal{A}} = \{0, 1, \ldots, 2(q-1)\}$. From Lemma 6.7 (with alphabet $\hat{\mathcal{A}}$), the mod $2q-1$ sum target function
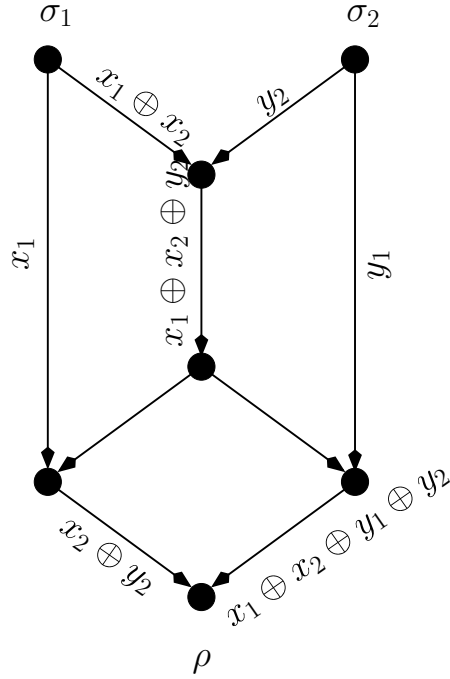
Figure 6: The reverse butterfly network with a code that computes the mod q sum target function.

can be computed in $\mathcal{N}$ at rate 2. Indeed for every $n \geq 1$, there exists a $(2n, n)$ network code that computes the mod $2q - 1$ sum target function at rate 2. So for the remainder of this proof, let $k = 2n$. Furthermore, every such code using $\hat{\mathcal{A}}$ can be "simulated" using $\mathcal{A}$ by a corresponding $\left(2n, \lceil n \log_q (2q - 1) \rceil\right)$ code for computing the mod $2q - 1$ sum target function, as follows. Let n' be the smallest integer such that $q^{n'} \geq (2q-1)^n$, i.e., $n' = \lceil n \log_q (2q - 1) \rceil$. Let $g : \hat{\mathcal{A}}^n \to \mathcal{A}^{n'}$ be an injection (which exists since $q^{n'} \geq (2q-1)^n$) and let the function g^{-1} denote the inverse of g on it's image $g(\hat{\mathcal{A}})$. Let $x^{(1)}, x^{(2)}$ denote the first and last, respectively, halves of the message vector $\alpha(\sigma_1) \in \mathcal{A}^{2n}$, where we view $x^{(1)}$ and $x^{(2)}$ as lying in $\hat{\mathcal{A}}^n$ (since $\mathcal{A} \subseteq \hat{\mathcal{A}}$). The corresponding vectors $y^{(1)}, y^{(2)}$ for the source σ_2 are similarly defined.

Figure 7 illustrates a $(2n, n')$ code for network $\mathcal{N}$ using alphabet $\mathcal{A}$ where '$\oplus$' denotes the mod $2q-1$ sum. Each of the nodes in $\mathcal{N}$ converts each of the received vectors over $\mathcal{A}$ into a vector over $\hat{\mathcal{A}}$ using the function g^{-1}, then performs coding in Figure 6 over $\hat{\mathcal{A}}$, and finally converts the result back to $\mathcal{A}$. Similarly, the receiver node T computes the component-wise arithmetic sum of the source message vectors $\alpha(\sigma_1)$ and $\alpha(\sigma_2)$ using

$$\begin{aligned}
&\alpha(\sigma_1) + \alpha(\sigma_2) \\
&= \big(g^{-1}(g(x^{(1)} \oplus x^{(2)} \oplus y^{(1)} \oplus y^{(2)})) \ominus g^{-1}(g(x^{(2)} \oplus y^{(2)})), \\
&\quad\quad g^{-1}(g(x^{(2)} \oplus y^{(2)}))\big) \\
&= (x^{(1)} \oplus y^{(1)}, x^{(2)} \oplus y^{(2)}).
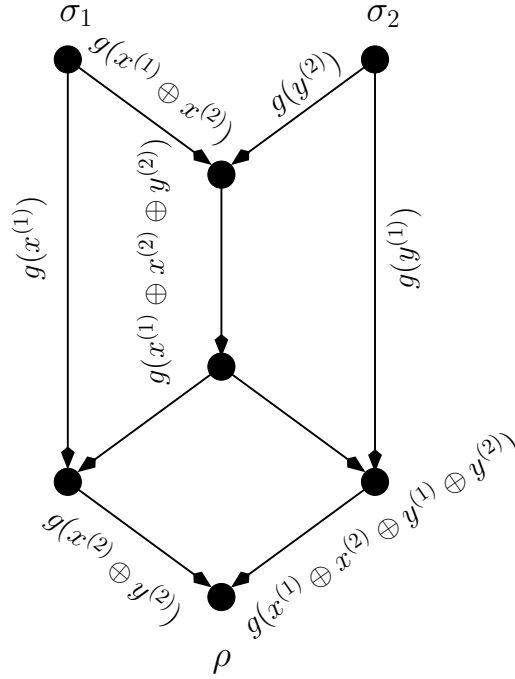\end{aligned}$$

Figure 7: The reverse butterfly network with a code that computes the arithmetic sum target function. '$\oplus$' denotes $\mod 2q - 1$ addition.

For any $n \geq 1$, the above code computes the arithmetic sum target function in $\mathcal{N}$ at a rate of

$$\frac{k}{n'} = \frac{2n}{\lceil n \log_q (2q - 1) \rceil}.$$

Thus for any $\epsilon > 0$, by choosing n large enough we obtain a code that computes the arithmetic sum target function, and which achieves a computing rate of at least

$$\frac{2}{\log_q (2q - 1)} - \epsilon.$$

■

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Transactions on Information Theory*, vol. IT-46, no. 4, pp. 1204–1216, July 2000.

[2] N. Alon, "Combinatorial Nullstellensatz", *Combinatorics, Probability and Computing*, no. 8, pp. 7-29, 1999.

[3] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network computing capacity for the reverse butterfly network", *IEEE International Symposium on Information Theory (ISIT)*, pp. 259-262, 2009.

[4] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds", *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1015-1030, February 2011.

[5] J. Cannons, R. Dougherty, C. Freiling, and K. Zeger, "Network routing capacity", *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 777-788, March 2006.

[6] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow", *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2745-2759, August 2005.

[7] R. Dougherty, C. Freiling, and K. Zeger, "Linear network codes and systems of polynomial equations", *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2303-2316, May 2008.

[8] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks", *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98-107, 2006.

[9] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, 1979.

[10] N. J. A. Harvey, R. D. Kleinberg, and A. Rasala Lehman, "Comparing network coding with multicommodity flow for the k-pairs communication problem", *MIT LCS Technical Report 964*, 2004.

[11] N. J. A. Harvey, R. Kleinberg, and A. Rasala Lehman, "On the capacity of information networks", *IEEE Transactions on Information Theory & IEEE/ACM Transactions on Networking (joint issue)*, vol. 52, no. 6, pp. 2345-2364, 2006.

[12] R. Hill, "A First Course in Coding Theory", Oxford University Press, 1990.

[13] K. M. Hoffman and R. Kunze, "Linear Algebra", Prentice Hall, 1971.

[14] R. Koetter and M. Médard, "An algebraic approach to network coding", *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[15] H. Kowshik and P. R. Kumar, "Zero-error function computation in sensor networks", *Proceedings of the IEEE Conference on Decision and Control*, pp. 3787-3792, 2009.

[16] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding", *IEEE Transactions on Information Theory*, vol. 49, pp. 371-381, 2003.

[17] B. Nazer and M. Gastpar, "Computing over multiple-access channels", *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3498-3516, 2007.

[18] G. Nebe, E. M. Rains, and N. J. A. Sloane, "Self-Dual Codes and Invariant Theory", Springer, 2006.

[19] J. Paek, B. Greenstein, O. Gnawali, K. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The tenet architecture for tiered sensor networks", *ACM Transactions on Sensor Networks*, 2009.

[20] B. K. Rai, and B. K. Dey, "Sum-networks: System of polynomial equations, unachievability of coding capacity, reversibility, insufficiency of linear network coding", available at *http://arxiv.org/abs/0906.0695*, 2009.

[21] B. K. Rai, B. K. Dey, S. Shenvi, "Some bounds on the capacity of communicating the sum of sources", *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, 2010.

[22] A. Ramamoorthy, "Communicating the sum of sources over a network", *IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, 2008.

[23] A. Rasala Lehman and E. Lehman, "Complexity classification of network information flow problems", *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 142–150, 2003.